Background neural networks tutorial

Jarek van Dijk

20-03-2020

Center for Molecular
and Biomolecular Informatics (CMBI)

Institute for Molecular Life Sciences
Radboudumc

Image: quantamagazine.org

Teaching goals

1. Basic understanding of neural networks
2. Get familiar with terms
3. Get some hand on experience (2nd presentation)

Content

- Software
- Building blocks
  - Neurons
  - Loss functions
  - Back propagation
- Practicalities
- Architectures
- Problems / questions

Software

- **Software**
- Building blocks
  - Neurons
  - Loss functions
  - Back propagation
- Practicalities
- Architectures
- Problems / questions

Software

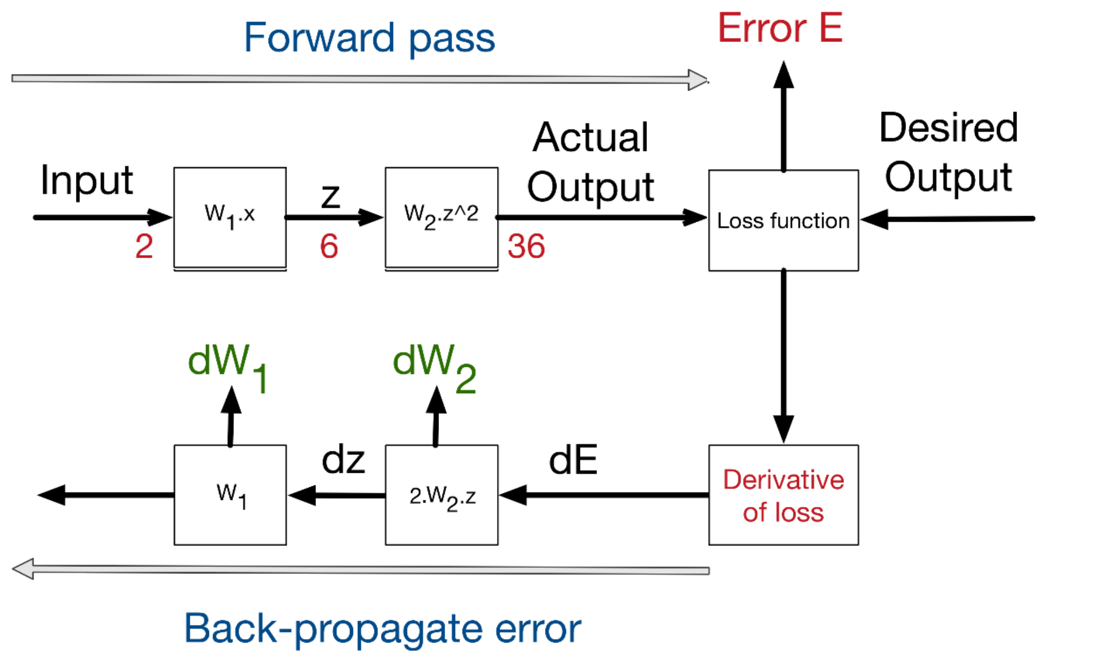
- TensorFlow
- Keras
- **Pytorch**

Building blocks

- Software
- **Building blocks**
  - Neurons
  - Loss functions
  - Back propagation
- Practicalities
- Architectures
- Problems / questions

Building blocks

- Neurons / networks
- Loss function
- Back propagation



Recap: vectors

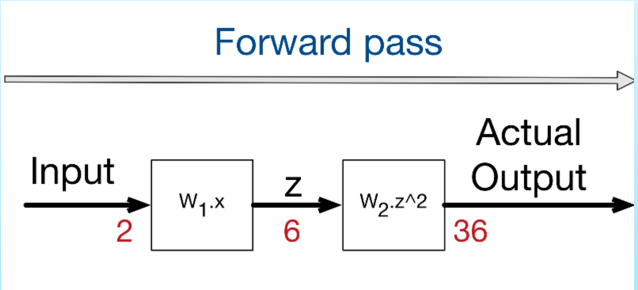
- Row times column
- Vectors to matrix

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \times \begin{bmatrix} 6 & 4 & 3 \end{bmatrix} = \begin{bmatrix} 12 & 8 & 6 \\ 18 & 12 & 9 \\ 24 & 16 & 12 \end{bmatrix}$$

Building blocks

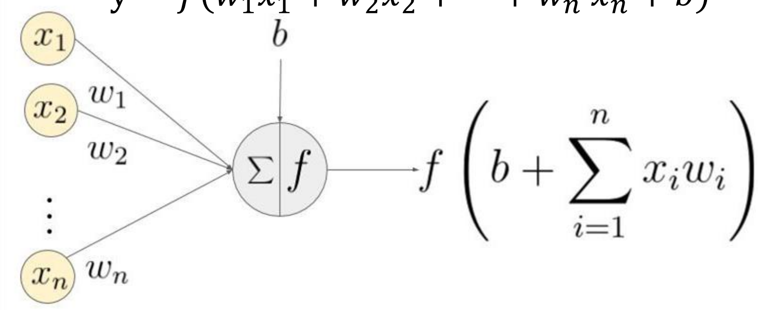
- Software
- Building blocks
  - **Neurons**
  - Loss functions
  - Back propagation
- Practicalities
- Architectures
- Problems / questions

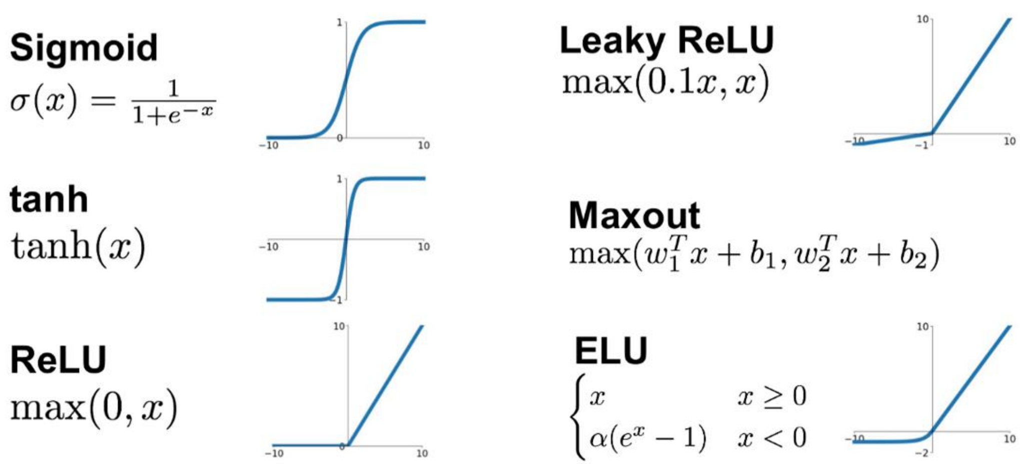


Building block: neuron

- Data transformer: input → output
  - Activation function (Input * Weight + Bias)
- Vectors → $Y = f(X \cdot W + b)$
- Non-linear

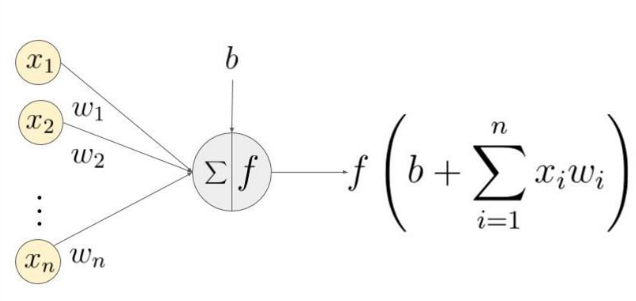
$$y = f(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b)$$

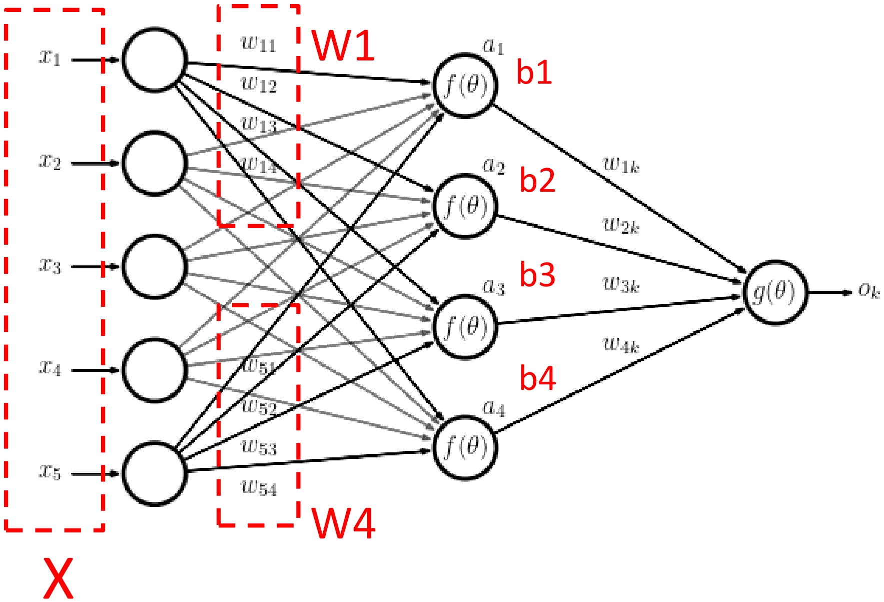




Building block: Neural Network

- Network of neurons



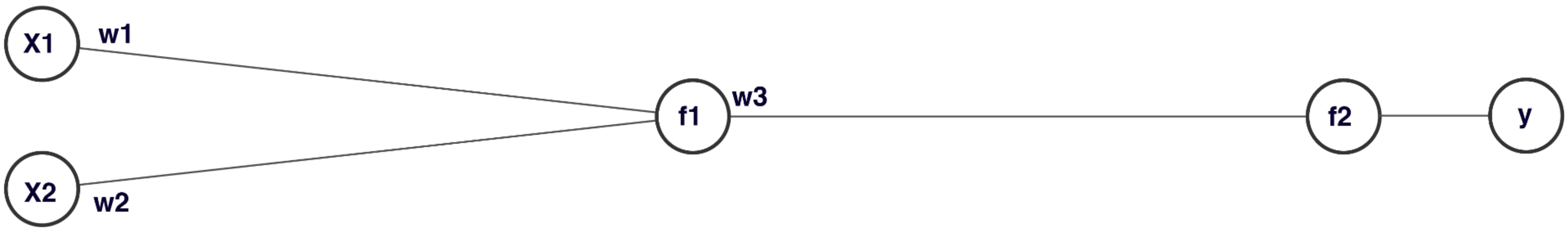


Neural Network – example1

$$y = f_2(\ f_1(\ x_1 w_1 + x_2 w_2 + \ b_1\ )w_3 + b_2)$$

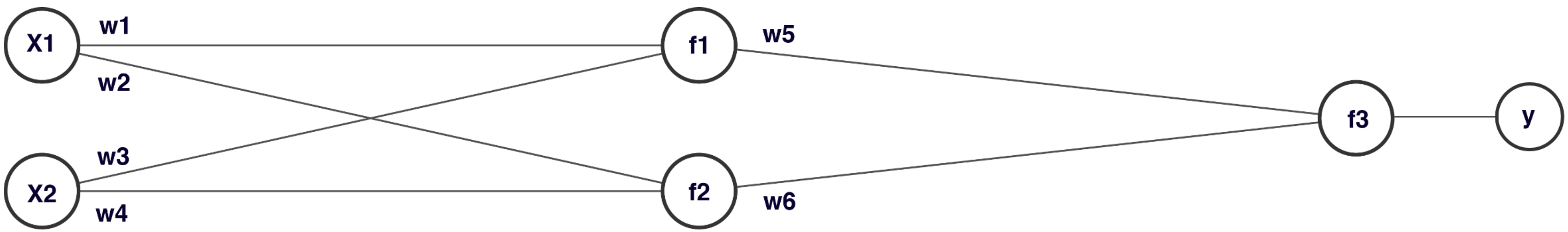
$$y = f_2(f_1(WX + B_1)w_3 + B_2)$$

$W = [w_1, w_2], X = [x_1; x_2]$



Neural Network – example2

$$y = f_3(\ f_1(\ x_1w_1 + x_2w_2 + \ b_1\ )w_5 + f_1(\ x_1w_3 + x_2w_4 + \ b_2\ )w_6 + \ b_3)$$
$$y = f_3(f_1(X_1W_1 + B_1)w_5 + f_1(XW_2 + B_2)w_6 + B_3)$$
$W_1[w_1, w_2]$, $\quad X = [x_1; x_2]$, $\quad W_2 = [w_3, w_4]$

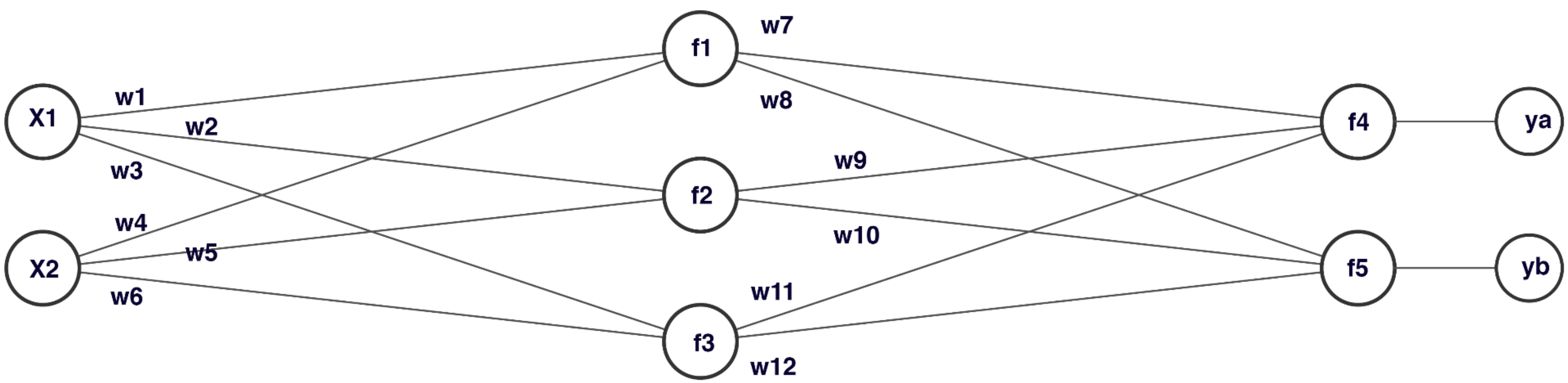


Neural Network – example3

$$y_a = f_4\left(\begin{array}{c} f_1(\,XW_1 +\ B_1\,)w_7 + f_2(\,XW_2 +\ B_2\,)w_9 + \\ f_3(\,XW_3 + B_3)w_{11} + B_4 \end{array}\right)$$

$$y_b =\ f_5\left(\begin{array}{c} f_1(\,XW_1 +\ B_1\,)w_8 + f_2(\,XW_2 +\ B_2\,)w_{10} + \\ f_3(\,XW_3 + B_3)w_{12} + B_4 \end{array}\right)$$

$X = [x_1, x_2],\qquad W_1 = [w_1; w_4],\qquad W_2 = [w_2; w_5],\qquad W_3 = [w_3, w_6]$



Building blocks

- Software
- Building blocks
  - Neurons
  - **Loss functions**
  - Back propagation
- Practicalities
- Architectures
- Problems / questions

Loss function

- Compare output to wanted output (just error measurement)
- Mean Squared Error: $\left(y_{pred} - y\right)^2$
- Cross Entropy: $y \cdot log\left(y_{pred}\right)$
- Custom ones

Building blocks

- Software
- Building blocks
  - Neurons
  - Loss functions
  - **Back propagation**
- Practicalities
- Architectures
- Problems / questions

Back propagation

- Relate loss to weights
- Differentiate loss on weights
  - Partial derivatives

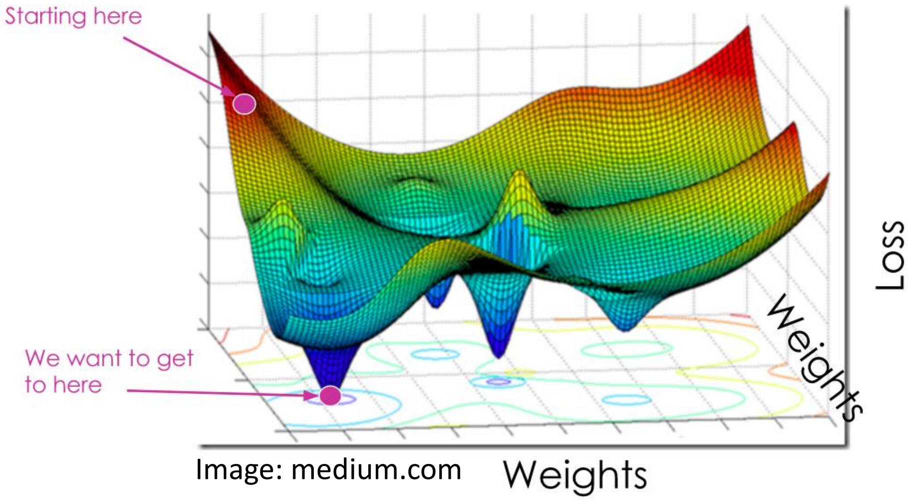


Image: medium.com

Recap: Differentiation

Differentiation Rules	
Constant Rule	$\frac{d}{dx}[c]=0$
Power Rule	$\frac{d}{dx}x^n = nx^{n-1}$
Product Rule	$\frac{d}{dx}[f(x)g(x)]=f'(x)g(x)+f(x)g'(x)$
Quotient Rule	$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right]=\frac{g(x)f'(x)-f(x)g'(x)}{[g(x)]^2}$
Chain Rule	$\frac{d}{dx}[f(g(x))]=f'(g(x))g'(x)$

$$\frac{dy}{dx} = \frac{dy}{du}\cdot\frac{du}{dv}\cdot\frac{dv}{dx}$$

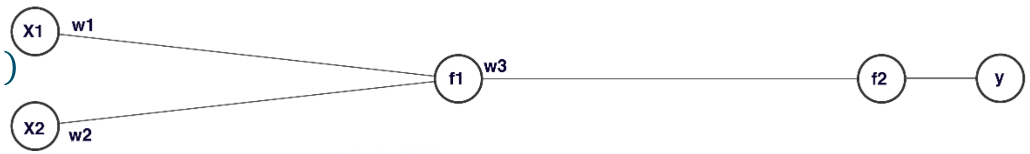
$$f(x) = (2x-2)^2$$

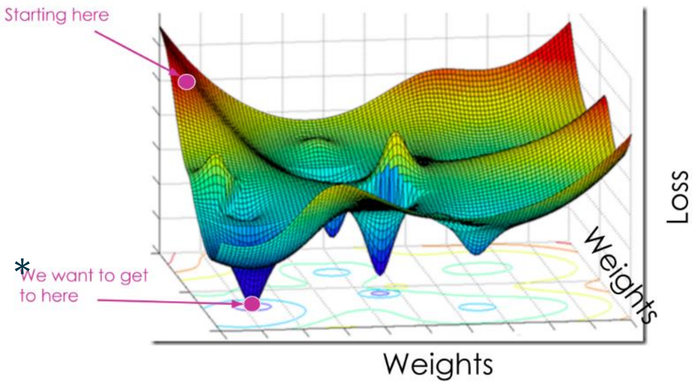
$$f(x) = g(u), \qquad u = 2x-2, \qquad g(u) = u^2$$

$$\frac{\partial f}{\partial x} = \frac{\partial g}{\partial u} * \frac{\partial u}{\partial x} = 2u * 2 = 4(2x-2)$$

Backpropagation - example

- $y_{pred} = f_2(\, f_1(\, x_1 w_1 + x_2 w_2 + \; b_1\,) w_3 + b_2)$
- $\text{loss} = \left(y \; - y_{pred}\right)^2$





Remember $f(x) = \; (2x - 2)^2$?

- $$\frac{\partial \mathrm{Loss}}{\partial \mathrm{w}_1} = 2\big(\mathrm{f}_1(\mathrm{f}_1(\mathrm{XW} + \; \mathrm{B}_1\,)\mathrm{w}_3 + \mathrm{B}_2)\big) * \\ -\mathrm{f}_2'(\mathrm{f}_1(\mathrm{XW} + \; \mathrm{B}_1\,)\mathrm{w}_3 + \mathrm{B}_2) * \\ \mathrm{f}_2'(\mathrm{XW} + \; \mathrm{B}_1\,)\mathrm{w}_3 * \mathrm{x}_1$$
- $$\frac{\partial \mathrm{Loss}}{\partial \mathrm{w}_2} = 2\big(\mathrm{f}_1(\mathrm{f}_1(\mathrm{XW} + \; \mathrm{B}_1\,)\mathrm{w}_3 + \mathrm{B}_2)\big) * \\ -\mathrm{f}_2'(\mathrm{f}_1(\mathrm{XW} + \; \mathrm{b}_1\,)\mathrm{w}_3 + \mathrm{B}_2) * \\ \mathrm{f}_2'(\mathrm{XW} + \; \mathrm{B}_1\,)\mathrm{w}_3 * \mathrm{x}_2$$
- $$\frac{\partial \mathrm{Loss}}{\partial \mathrm{w}_3} = 2(\mathrm{f}_1(\mathrm{XW} + \mathrm{B}_1\,)\mathrm{w}_3 + \mathrm{B}_2) \\ * -\mathrm{f}_2'(\mathrm{f}_1(\mathrm{XW} + \mathrm{B}_1\,)\mathrm{w}_3 + \mathrm{B}_2) \\ * \mathrm{f}_1(\mathrm{XW} + \mathrm{B}_1\,)$$

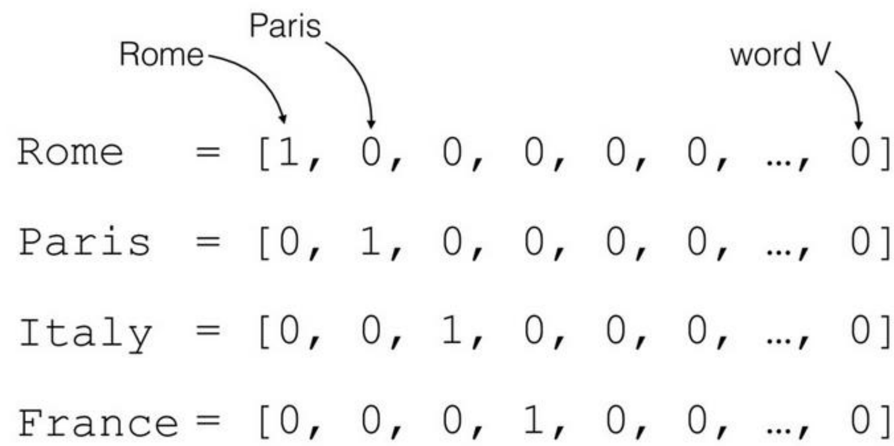
$$W = [w_1, w_2], \qquad X = [x_1; x_2]$$

Practicalities

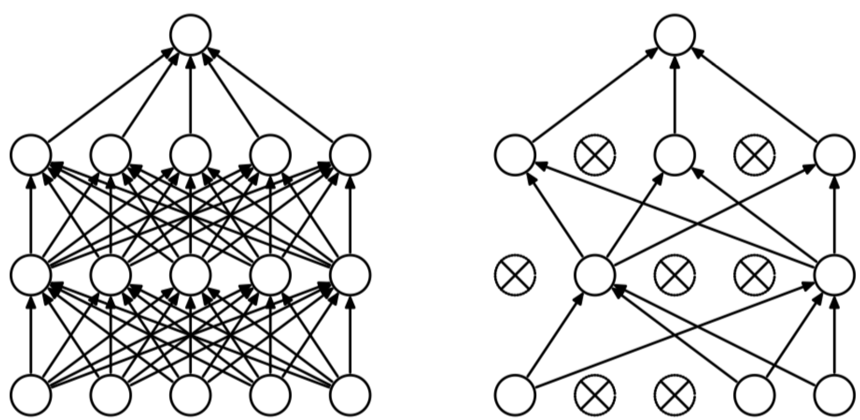
- Software
- Building blocks
  - Neurons
  - Loss functions
  - Back propagation
- **Practicalities**
- Architectures
- Problems / questions

Practicalities

- One hot encoding
- Dropout
- Batch normalization
- Padding



ARLMP --> ARLMP
ALP --> ALP * *
RNYY --> RNYY *



(a) Standard Neural Net (b) After applying dropout.

Architectures

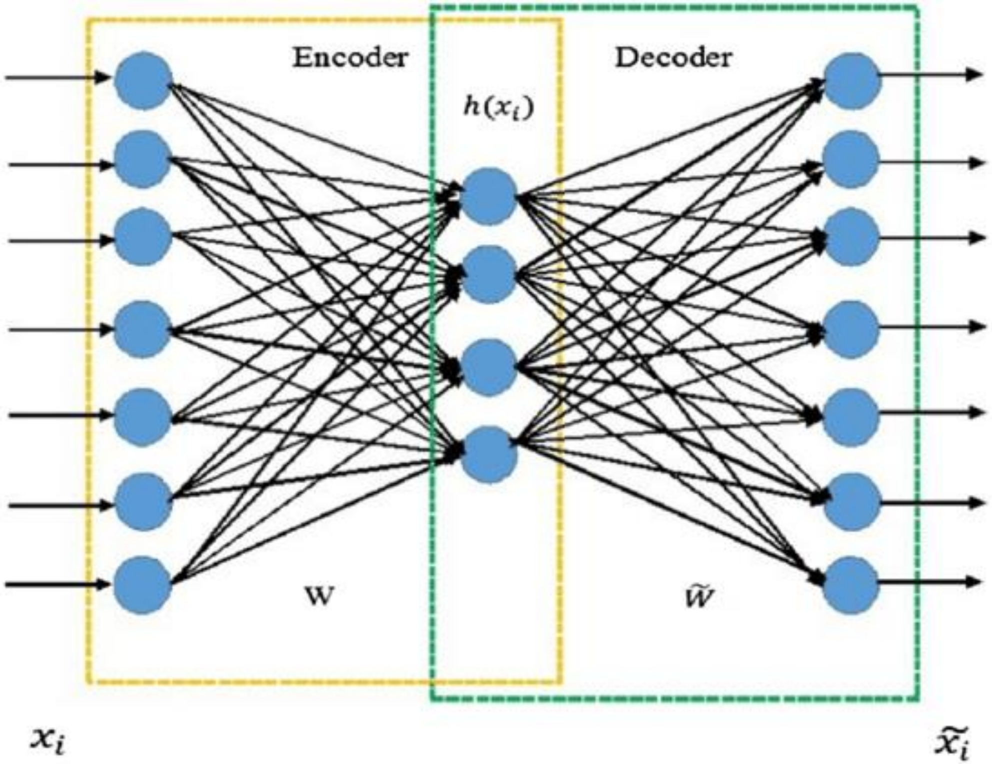
- Software
- Building blocks
  - Neurons
  - Loss functions
  - Back propagation
- Practicalities
- **Architectures**
- Problems / questions

Architectures

- Fully Connected
- Auto encoder
- Recurrent Neural Network (RNN)
- Convolutional Neural Network (CNN)
- Generative Adversarial Network (GAN)

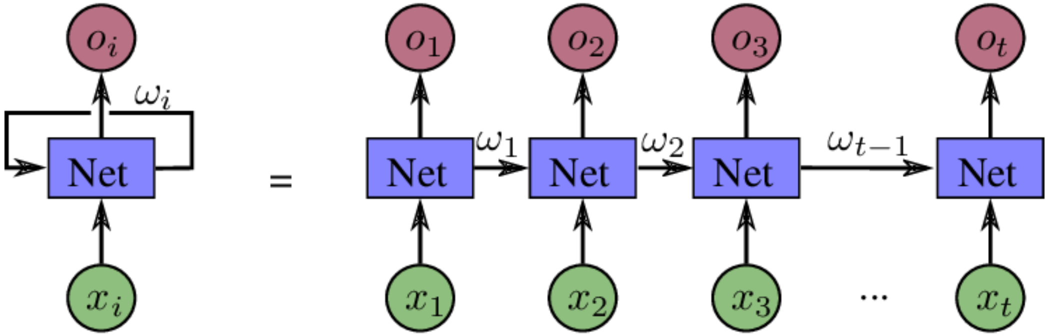
Auto encoder

- Input = output
- Compress data



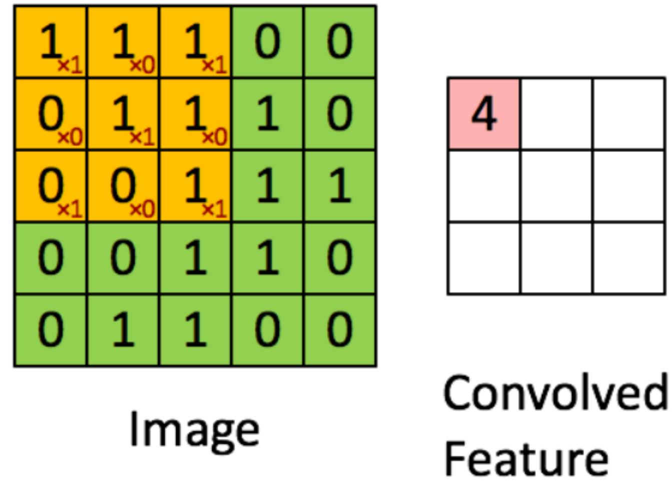
Recurrent neural network

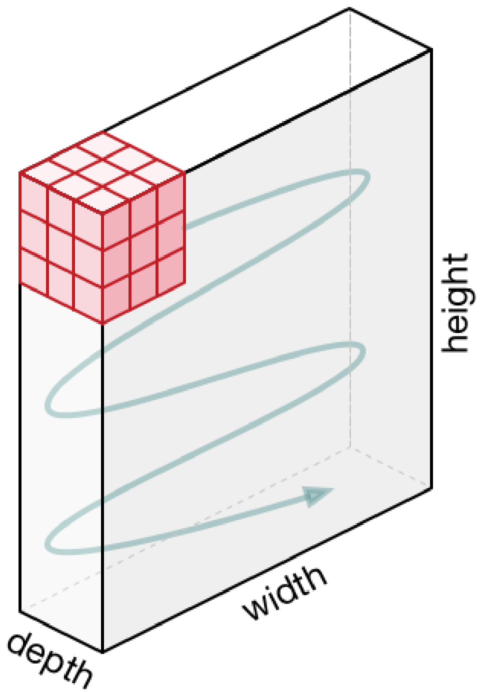
- Memory
- Variable input length
  - Same amount of features!
- Variable output length

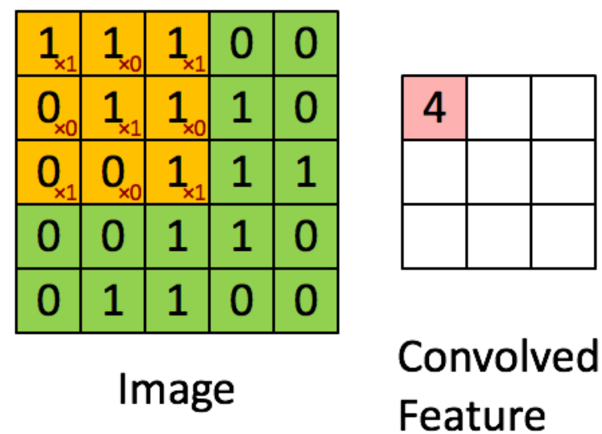


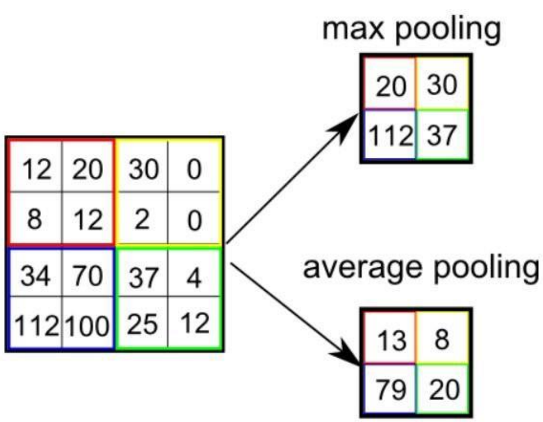
Convolutional neural network

- Windows
- Re-usable patterns
- Depth = amount of filters
- Pooling



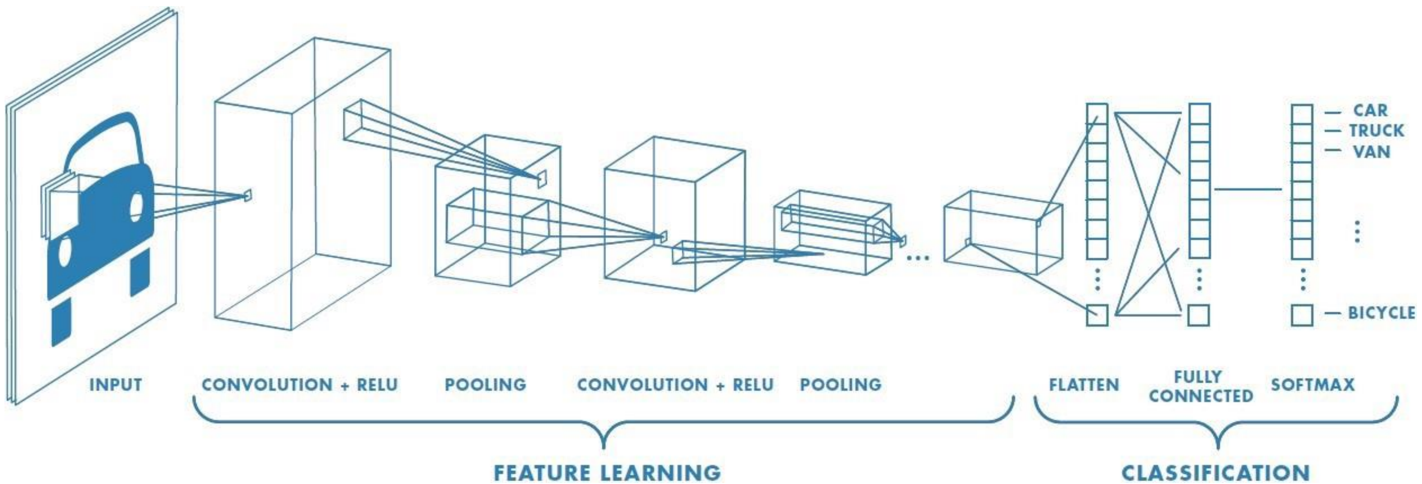






Convolutional neural network

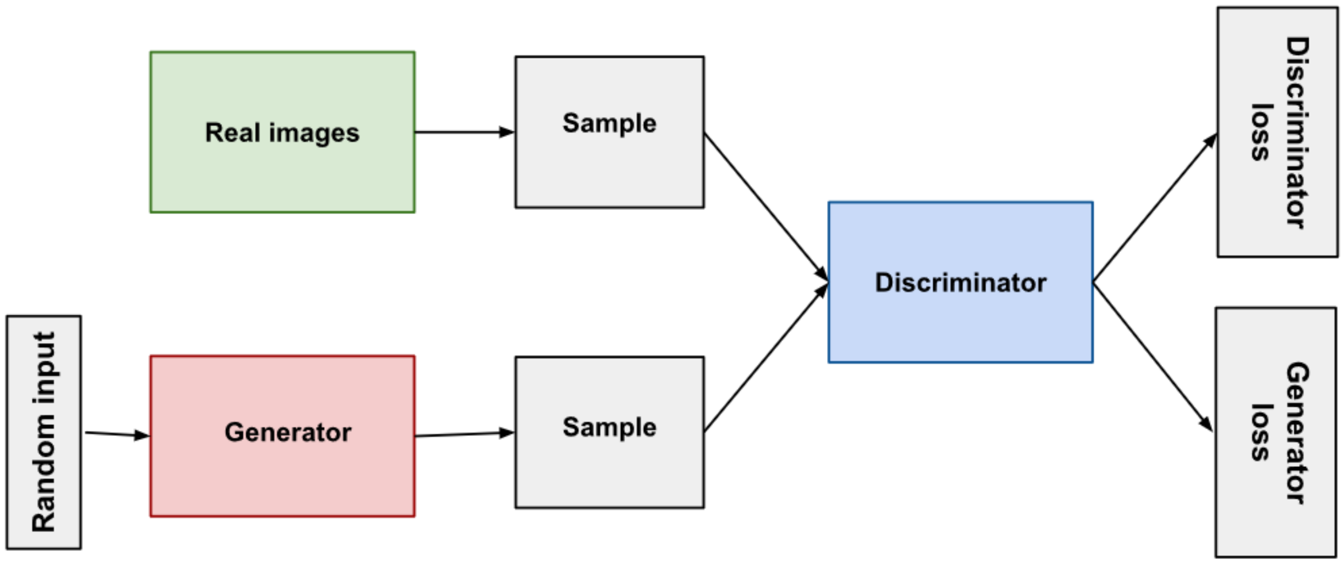
- Filters on filters
- New data representation



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

GANS

- Network vs network



Problems / questions

- Software
- Building blocks
  - Neurons
  - Loss functions
  - Back propagation
- Practicalities
- Architectures
- **Problems / questions**

Problems

- Get right input (represent features)
- Get good loss function (no problem vs youre welcome)
- Check output (no shortcuts? Overfitting?)
- Get good architecture
- **Need domain knowledge**

Further reading

Links I used:

- https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/
- https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec
- https://www.digitalvidya.com/blog/types-of-neural-networks/
- https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch
- https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- https://blog.floydhub.com/gru-with-pytorch/
- https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be
- https://medium.com/udacity-pytorch-challengers/a-brief-overview-of-loss-functions-in-pytorch-c0ddb78068f7
- https://playground.tensorflow.org/

Papers:

- "Convolutional Neural Network Architectures for Matching Natural Language Sentences"
- "Deep Visual-Semantic Alignments for Generating Image Descriptions"
- Papers by D.T. Rademaker

Questions